技术：D3.js

梅鸿辉

浙江大学 CAD&CG国家重点实验室

Data-Driven Documents

近年来，许多报刊杂志、新闻媒体、网站等大量使用可视化技术展示复杂的数据，方便读者理解数据所包含的信息。随之涌现出大量信息可视化工具，D3正是其中被最为广泛使用的工具之一。

D3简介 1

D3.js

- D3全称 Data-Driven Documents
- JavaScript函数库
- 用于在网页上绘制可视化

优缺点

- 优点
  - 功能丰富：相对底层，可以自由地编写
  - 代码优雅
  - 学习资料多：大量示例，还有其他开源封装
  - 配置简单：HTML、CSS和SVG
- 缺点
  - 学习曲线高
  - 性能问题

参考资料

- 源代码：https://github.com/d3/d3
- 官网：https://d3js.org/
- 中文wiki：https://github.com/d3/d3/wiki/API--中文手册
- 作者博客： http://bost.ocks.org/mike/ http://bl.ocks.org/mbostock/
- Dashing D3.js：https://www.dashingd3js.com/table-of-contents

- M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-Driven Documents. *IEEE Trans. Visualization & Comp. Graphics*, 17(12):2301–2309, 2011. (链接)

安装部署 2

预备知识

- 浏览器 (IE / EDGE / Chrome / Firefox / Safari / Opera)

- HTML (Hypertext Markup Language)
- CSS (Cascading Style Sheets)
- JavaScript
- DOM (Document Object Model)
- SVG (Scalable Vector Graphics)

- http://www.w3school.com.cn/

准备工具

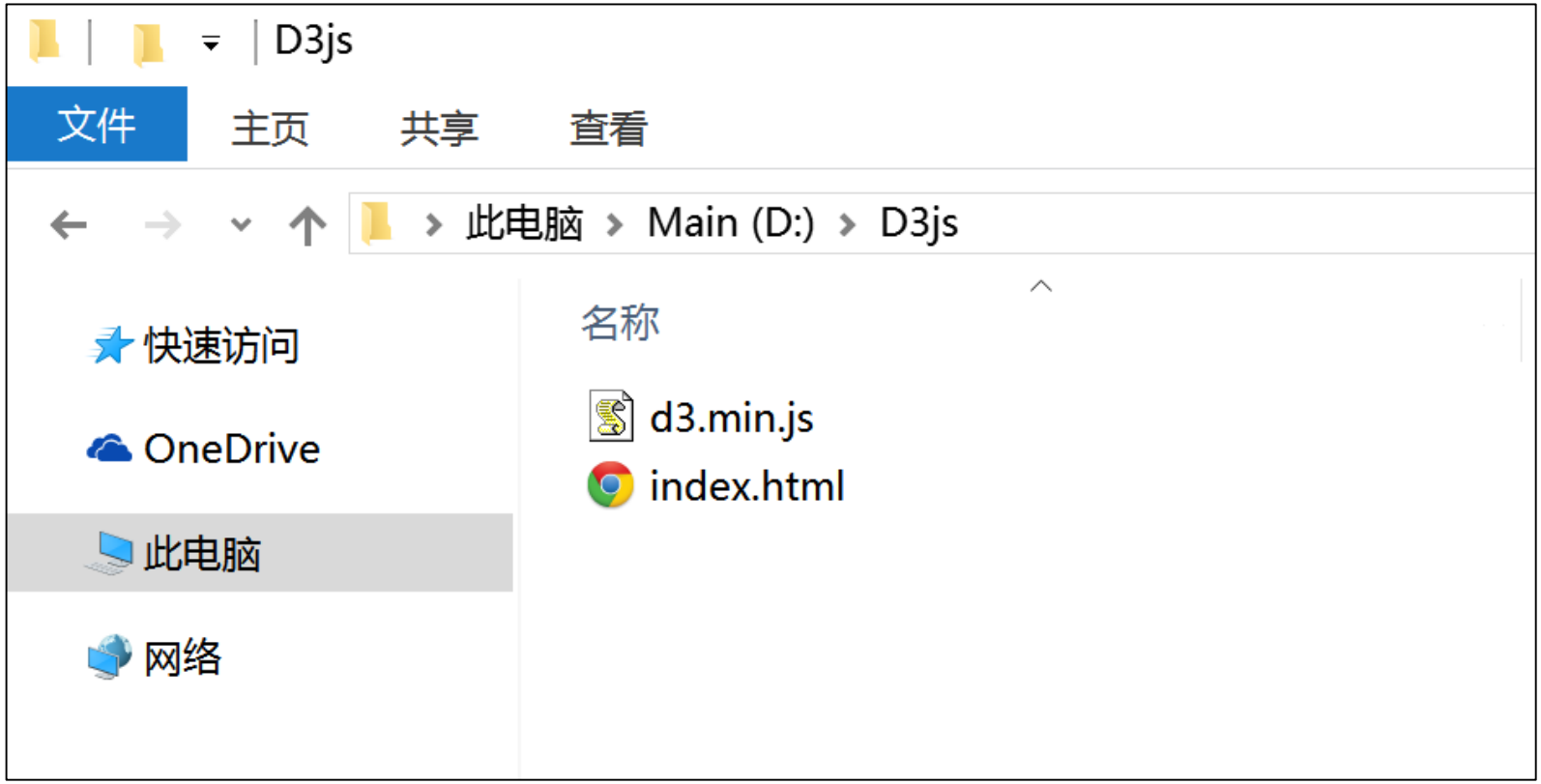
- 记事本软件：Notepad++、Sublime Text 等
- 浏览器：IE 9+ / EDGE / Chrome / Firefox / Safari / Opera，推荐用 Chrome
- 服务器软件：Apache、Tomcat 等

- 本次演讲使用：Sublime Text + Chrome + XAMPP

编写HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <script src="d3.min.js"></script>
    </head>
    <body>
        <p>Hello!</p>
    </body>
</html>
```

设置D3.js


D3js
文件
主页
共享
查看
此电脑 › Main (D:) › D3js
快速访问
OneDrive
此电脑
网络
名称
d3.min.js
index.html

设置D3.js

- 下载D3

https://github.com/d3/d3/releases/download/v4.1.1/d3.zip

- 包含.js文件

```
<script src="d3.min.js"></script>
```

或(调试时)：`<script src="d3.js"></script>`

- 直接引用

```
<script src="https://d3js.org/d3.v4.min.js"></script>
```

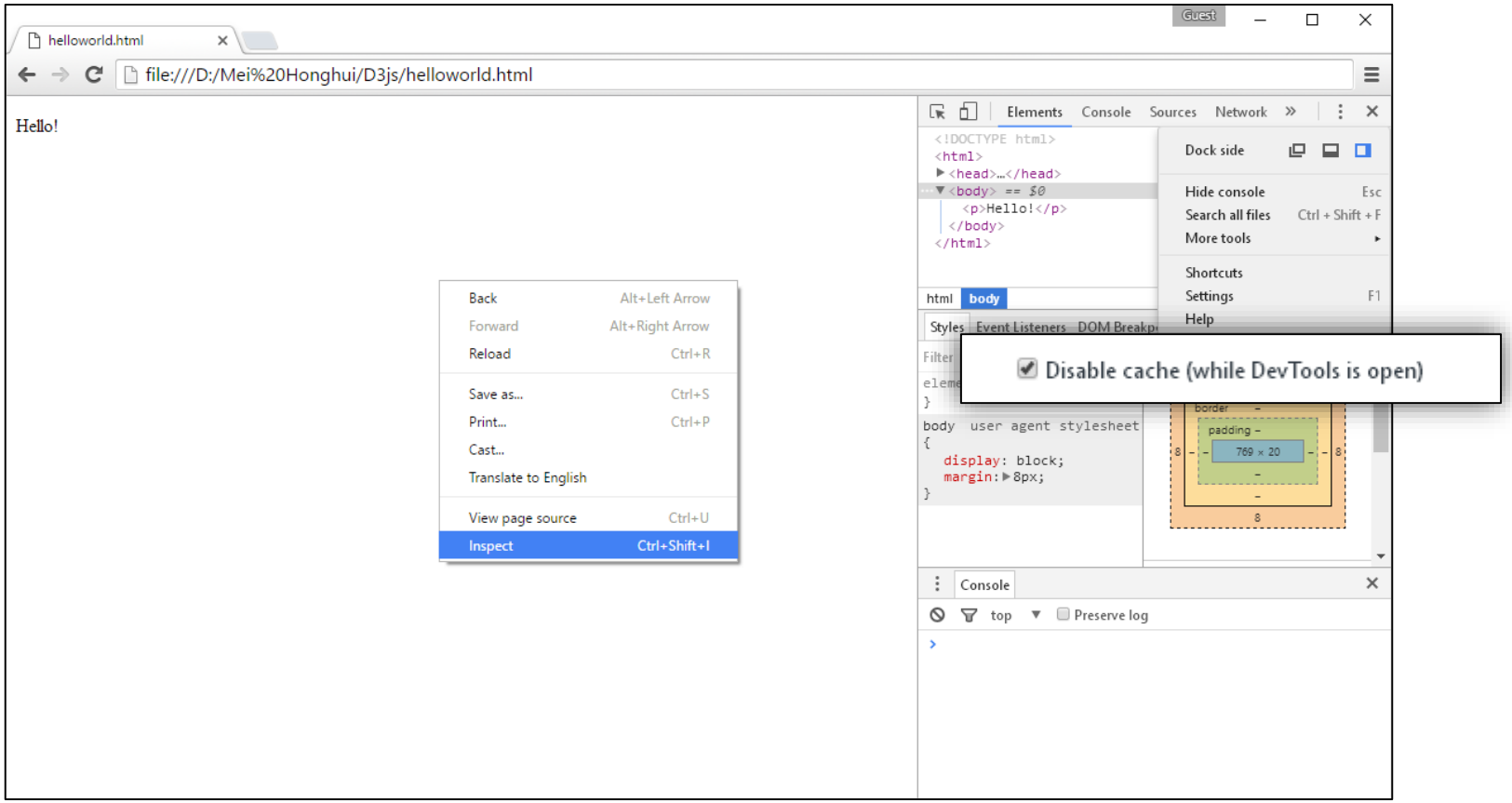
- V3（建议初学者使用）

https://github.com/d3/d3/releases/download/v3.5.17/d3.zip

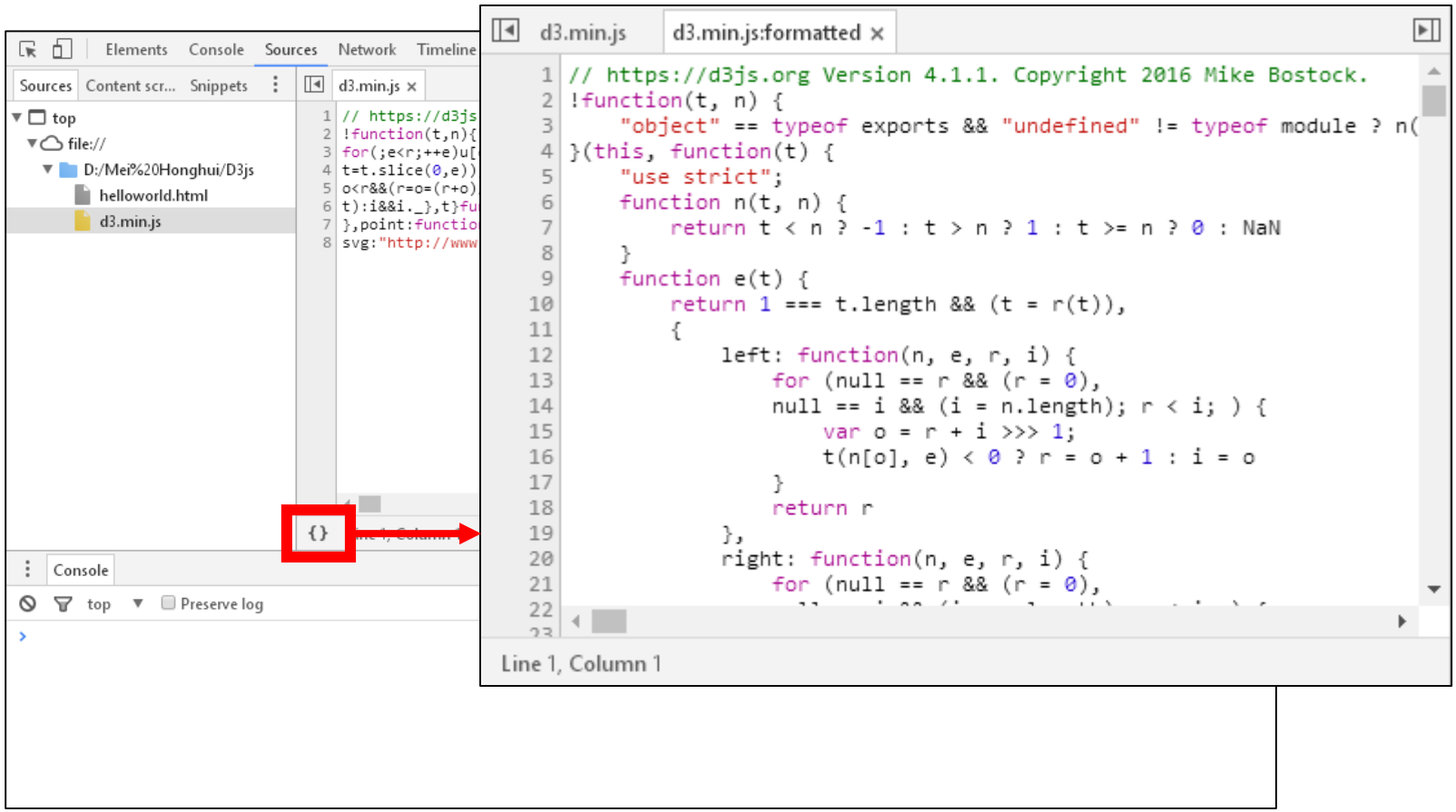
```
<script src="https://d3js.org/d3.v3.min.js"></script>
```

(v4与v3的区别： https://github.com/d3/d3/blob/master/CHANGES.md)

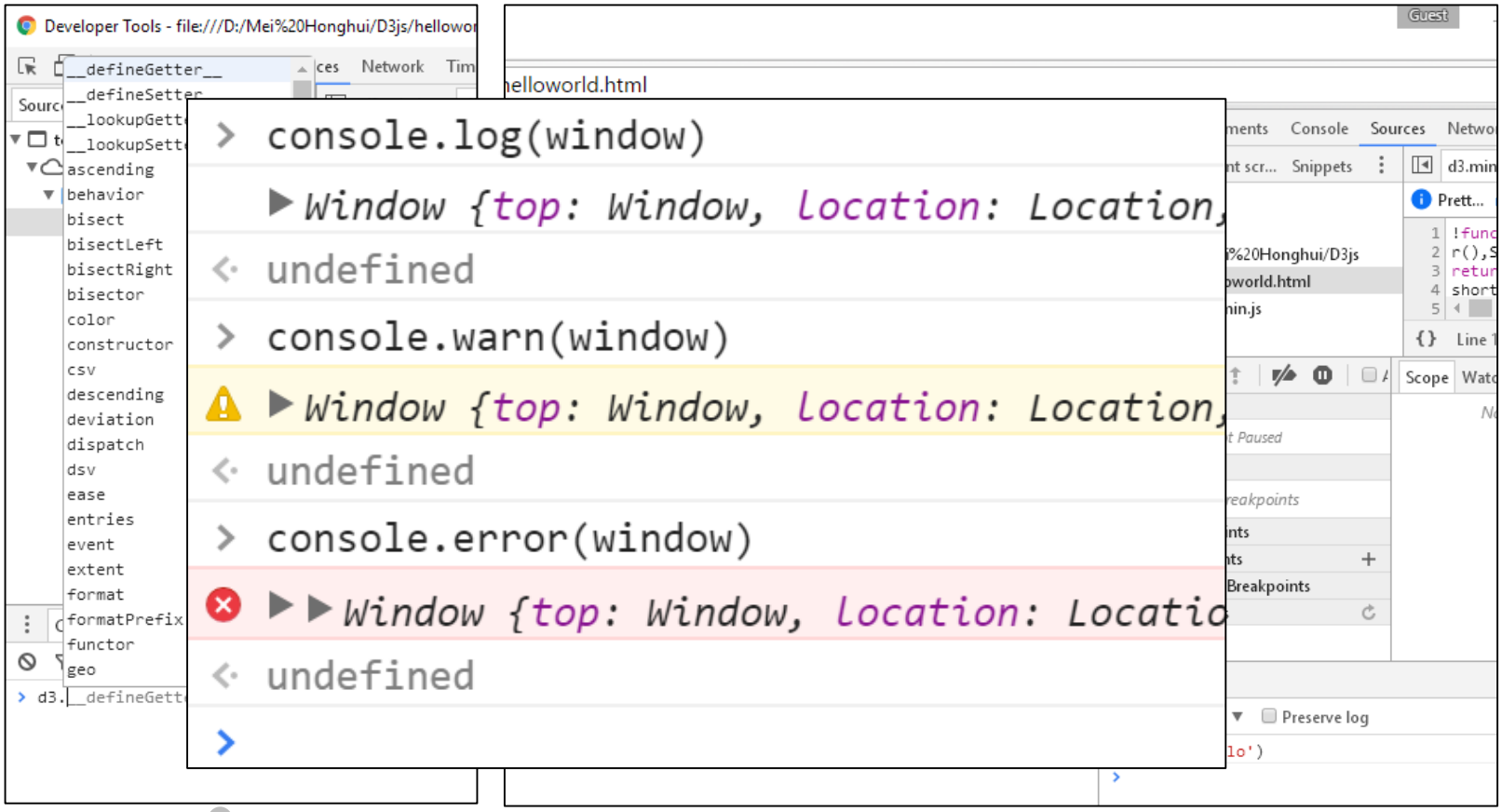
调试工具


helloworld.html
file:///D:/Mei%20Honghui/D3js/helloworld.html
Hello!
Back Alt+Left Arrow
Forward Alt+Right Arrow
Reload Ctrl+R
Save as... Ctrl+S
Print... Ctrl+P
Cast...
Translate to English
View page source Ctrl+U
Inspect Ctrl+Shift+I
Elements Console Sources Network
Dock side
Hide console Esc
Search all files Ctrl + Shift + F
More tools
Shortcuts
Settings F1
Help
Disable cache (while DevTools is open)
Console
top
Preserve log

调试工具

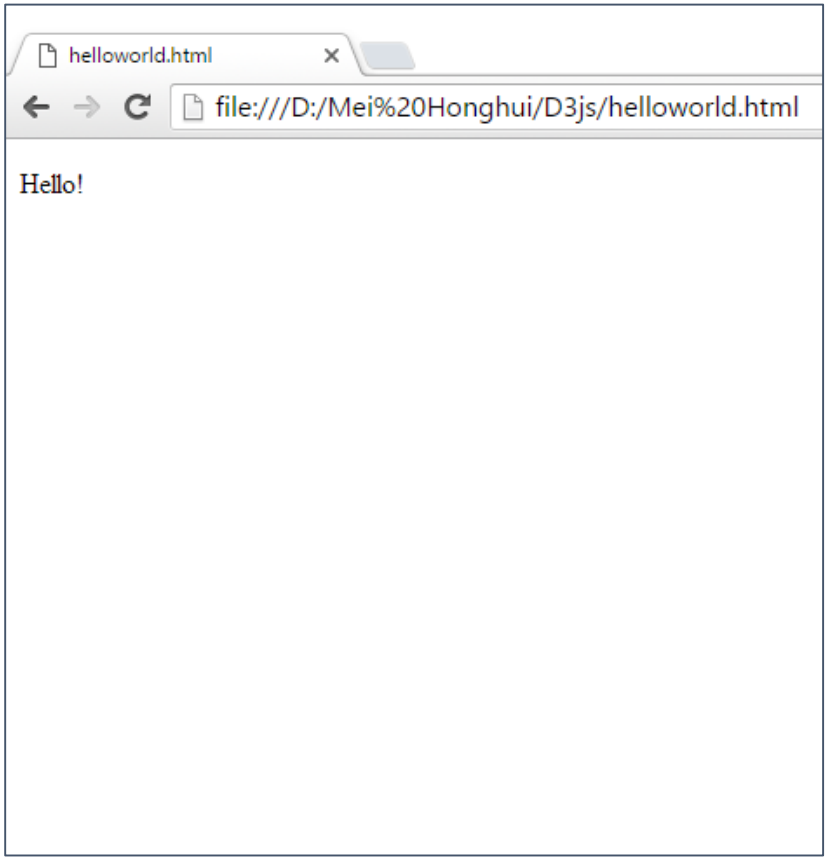

Elements
Console
Sources
Network
Timeline
Sources
Content scr...
Snippets
top
file://
D:/Mei%20Honghui/D3js
helloworld.html
d3.min.js
d3.min.js ×
{}
Console
top
Preserve log
d3.min.js
d3.min.js:formatted ×
// https://d3js.org Version 4.1.1. Copyright 2016 Mike Bostock.
!function(t, n) {
"object" == typeof exports && "undefined" != typeof module ? n(
}(this, function(t) {
"use strict";
function n(t, n) {
return t < n ? -1 : t > n ? 1 : t >= n ? 0 : NaN
}
function e(t) {
return 1 === t.length && (t = r(t)),
{
left: function(n, e, r, i) {
for (null == r && (r = 0),
null == i && (i = n.length); r < i;) {
var o = r + i >>> 1;
t(n[o], e) < 0 ? r = o + 1 : i = o
}
return r
},
right: function(n, e, r, i) {
for (null == r && (r = 0),
Line 1, Column 1

调试工具


Developer Tools - file:///D:/Mei%20Honghui/D3js/hellowor
__defineGetter__
__defineSetter__
__lookupGett
__lookupSett
ascending
behavior
bisect
bisectLeft
bisectRight
bisector
color
constructor
csv
descending
deviation
dispatch
dsv
ease
entries
event
extent
format
formatPrefix
functor
geo
Network
helloworld.html
Guest
Console
Sources
Snippets
Scope
Preserve log
console.log(window)
▶ Window {top: Window, location: Location,
undefined
console.warn(window)
▶ Window {top: Window, location: Location,
undefined
console.error(window)
▶ ▶ Window {top: Window, location: Locatio
undefined

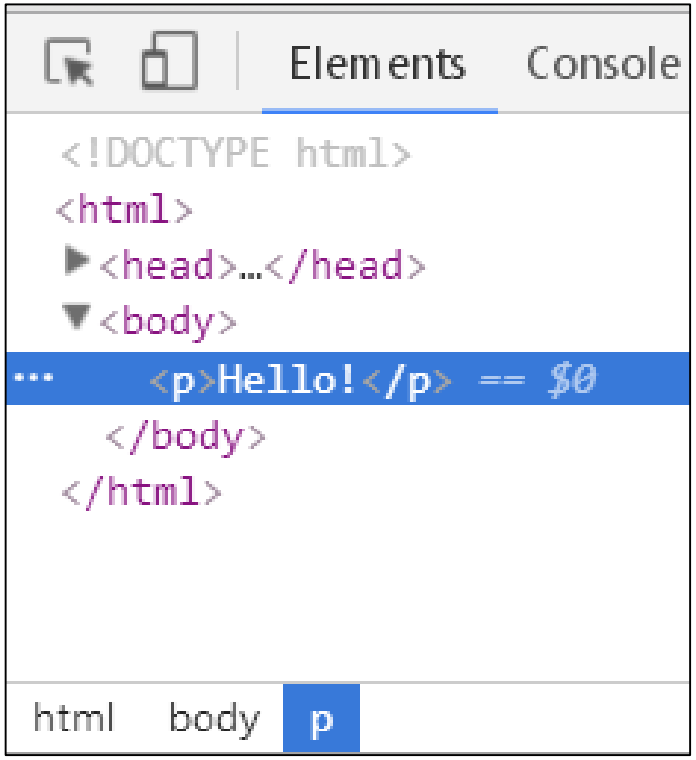
D3程序编写 3

Hello World




```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
                <script src="d3.min.js">
                </script>
        </head>
        <body>
                <p>Hello!</p>
        </body>
</html>
```

DOM元素

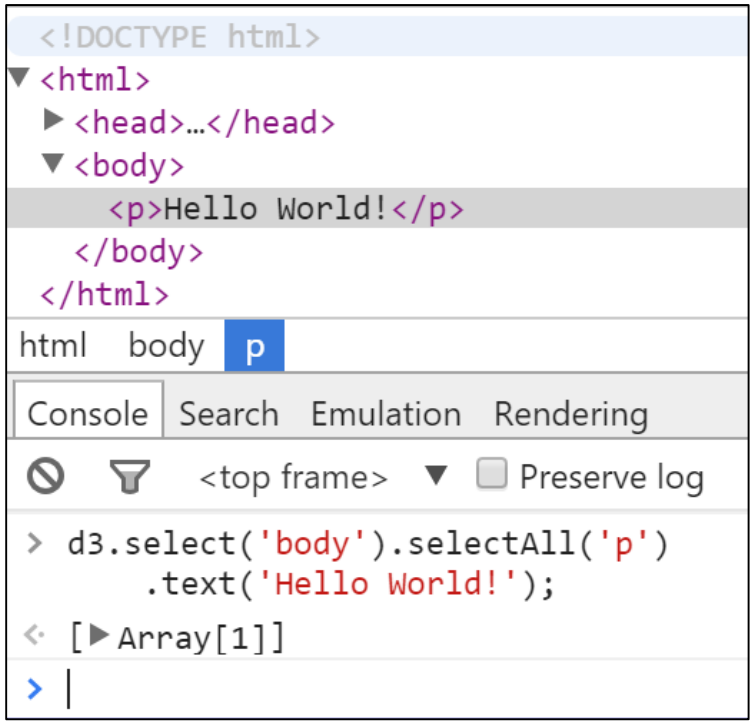


在命令行输入$0试试

```
> $0
    <p>Hello!</p>
> $0.appendChild
```

DOM元素

```
d3.select('body').selectAll('p').text('Hello World!');
```

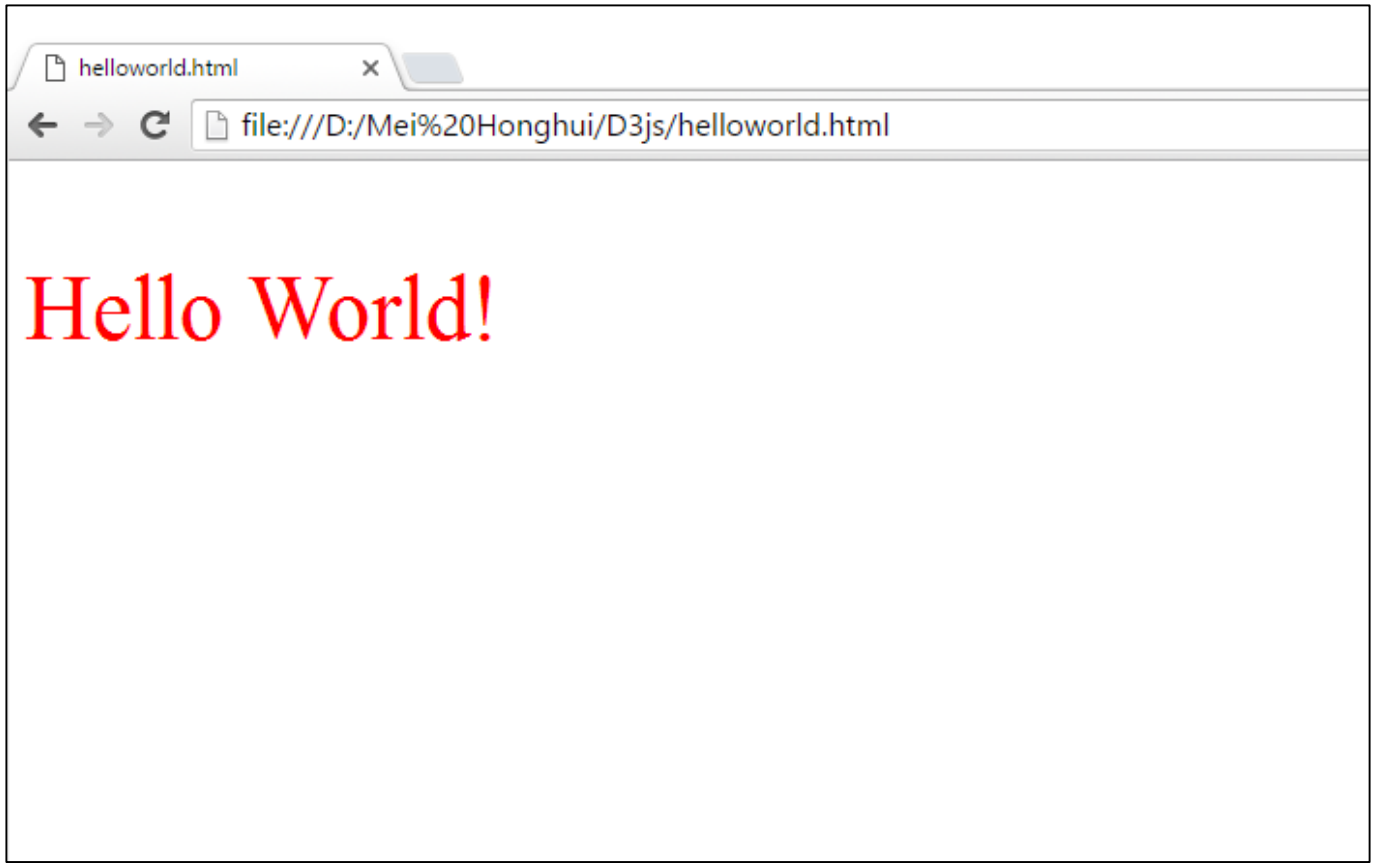


- select / selectAll
  - selector 选择器（i.e. 'body'）
  - selection 选择集

```
var p=d3.select('body').selectAll('p');
p.text('Hello World!');
p.style('color','red');
p.style('font-size','48px');
```

```
d3.select('body').selectAll('p')
  .text('Hello World!')
  .style('color','red')
  .style('font-size','48px');
```

DOM元素


helloworld.html
file:///D:/Mei%20Honghui/D3js/helloworld.html
Hello World!

补充知识：D3的缩进

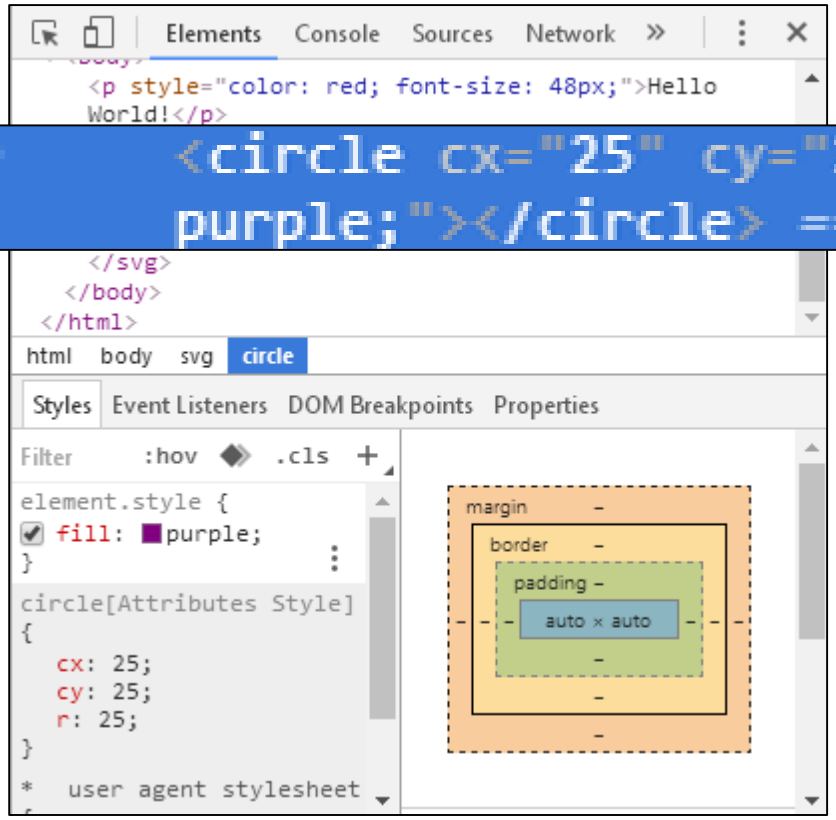
```
a=d3.select("body")
b=a.append("svg")
c=b.attr("width", width)
d=c.attr("height", height)
e=d.append("g")
f=e.attr("transform", "tran
g=f.append("rect")
...
```

```
▶ <body>…</body>
▶ <svg width="10" height="10">…</svg>
▶ <svg width="10" height="10">…</svg>
▶ <svg width="10" height="10">…</svg>
▶ <g transform="translate(100, 100)">…
▶ <g transform="translate(100, 100)">…
  <rect></rect>
```

绘制SVG

```
d3.select('body').append('svg')
    .attr('width', 50)
    .attr('height', 50)
.append('circle')
    .attr('cx', 25)
    .attr('cy', 25)
    .attr('r', 25)
    .style('fill', 'purple');
```

绘制SVG

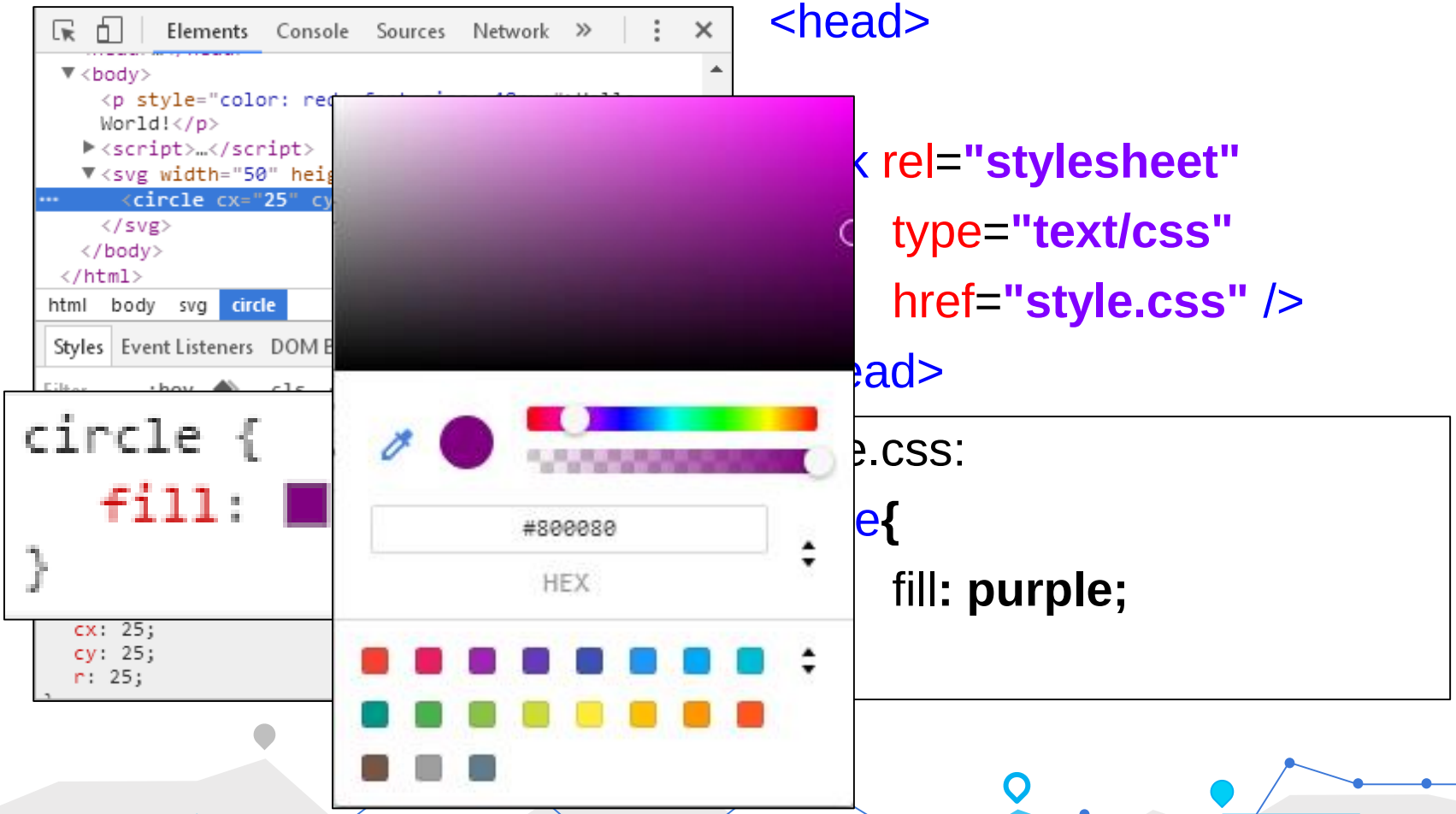



```
<head>
...
        type="text/css"
        href="style.css" />
</head>
```

style.css:

```
circle{
        fill: purple;
}
```

补充知识：CSS样式




```
<head>
    <link rel="stylesheet"
          type="text/css"
          href="style.css" />
</head>
```

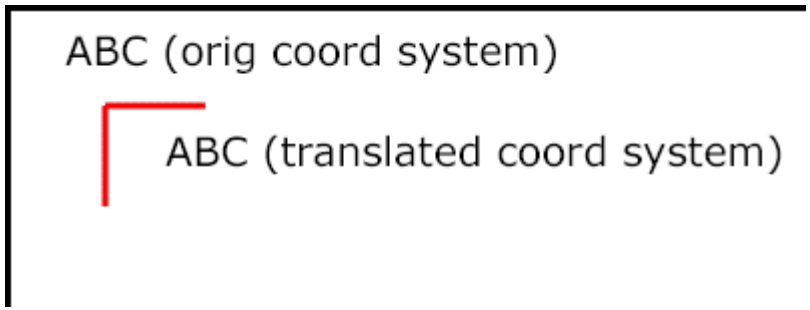
style.css:

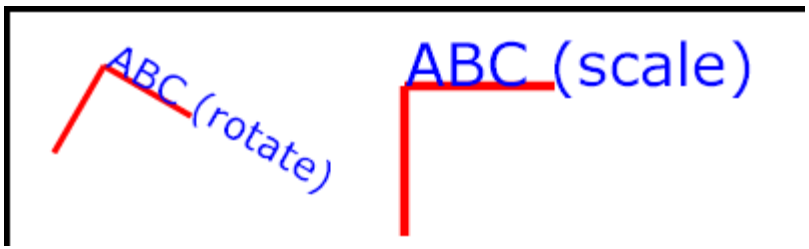
```
circle{
    fill: purple;
}
```

补充知识：SVG坐标系

- 原点在左上

- transform: translate / rotate / scale





scale

```
x=d3.scale.linear()
        .domain([min, max])
        .range([0, 300])
```

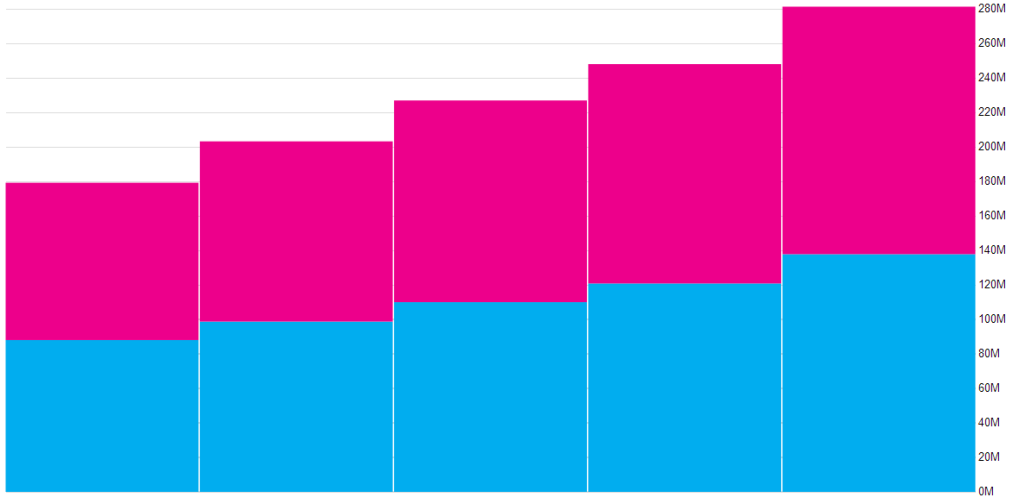
调用：x(10)

```
var min = d3.min(dataset);
var max = d3.max(dataset);
```

d3.scale.category10()

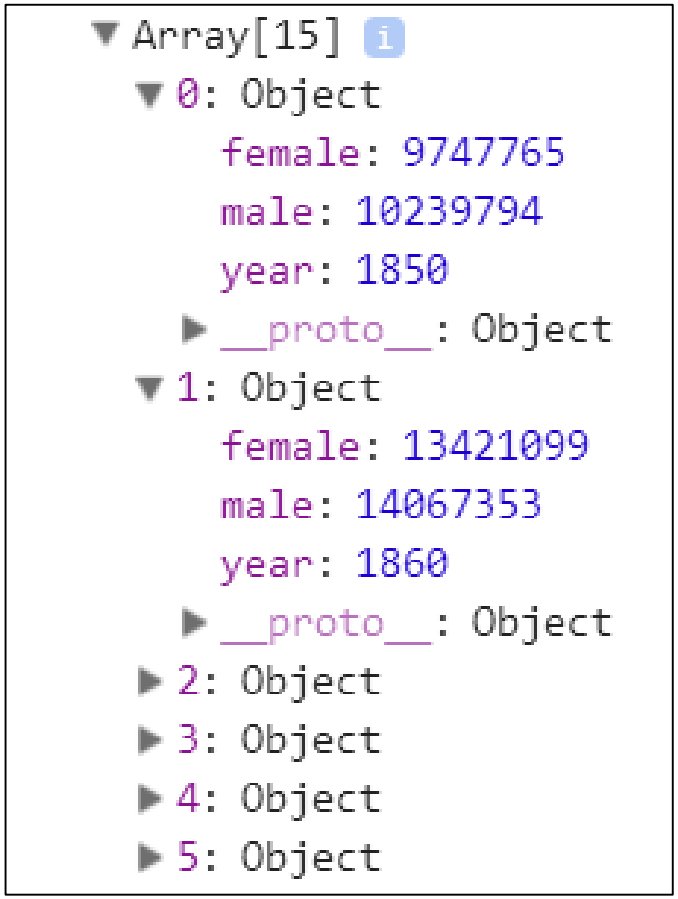
应用样例

year	male	female
1850	10239794	9747765
1860	14067353	13421099
1870	19401824	19120905
1880	25551807	24603241
1900	38915235	37347586
1910	47510999	44742351
1920	54149740	51871820
1930	61845657	60439516
1940	65950138	65718853
1950	74897011	75797823
1960	88075636	91225906
1970	98781173	104520833
1980	110040681	116981087
1990	120895364	127212264
2000	137863441	143557276

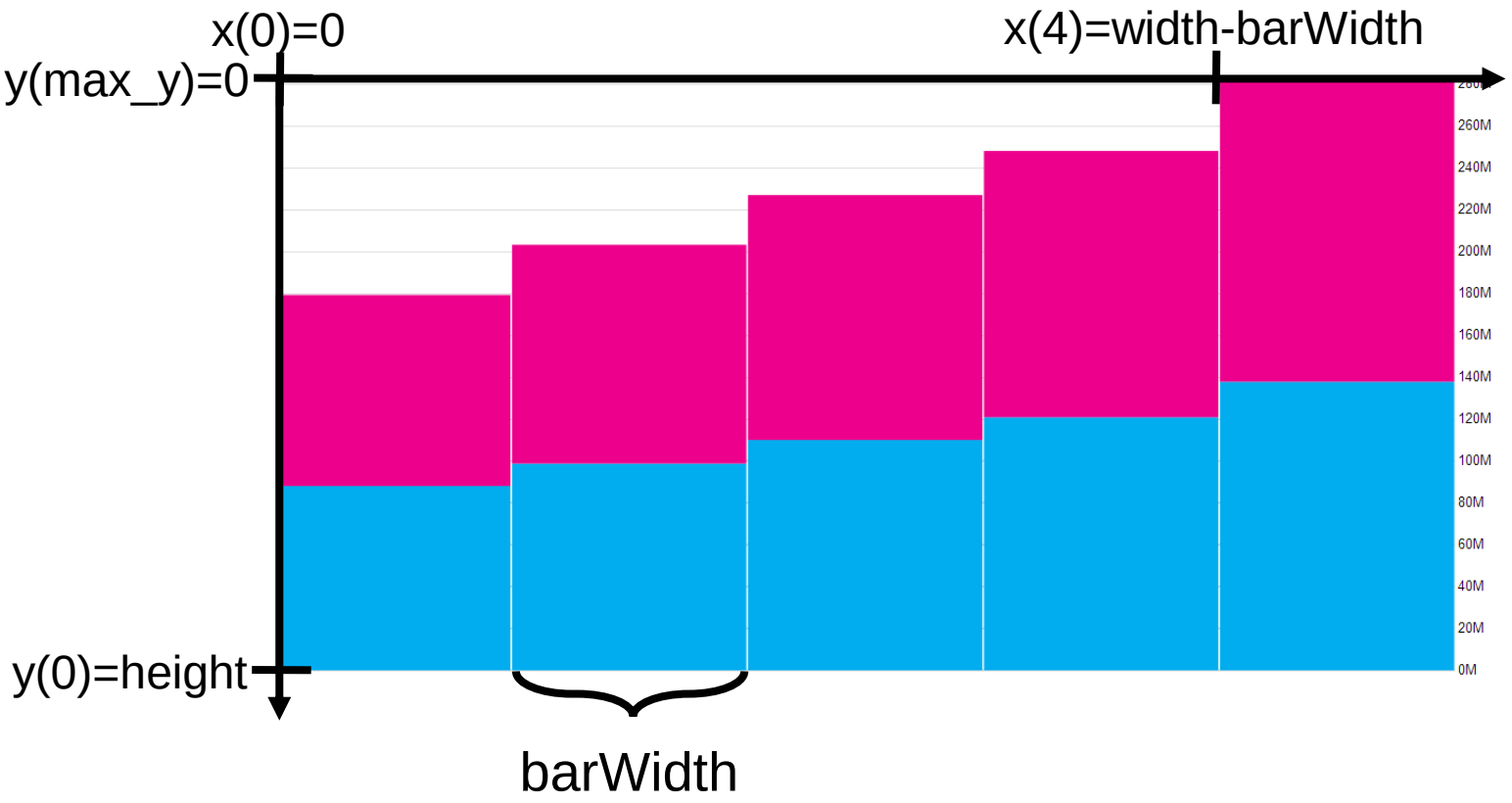


读取数据

```
d3.csv('population.csv',
    function(error,data){
        …
});
```



Step1：初始化


x(0)=0
x(4)=width-barWidth
y(max_y)=0
y(0)=height
barWidth
260M
240M
220M
200M
180M
160M
140M
120M
100M
80M
60M
40M
20M
0M

Step1：初始化

```
var x=d3.scale.linear()
    .domain([0,4])
    .range([0, width-barWidth]);
var y=d3.scale.linear()
    .range([height,0]);

var yAxis=d3.svg.axis()
    .scale(y)
    .orient("right")
    .tickSize(-width)
    .tickFormat(function(d){return Math.round(d/1e6)+"M";});
```

Step1：初始化

```
var svg = d3.select("body").append("svg")
        .attr("width", width+margin.left+margin.right)
        .attr("height", height+margin.top+margin.bottom)
.append("g")
        .attr("transform", "translate("+margin.left+","+margin.top+")");

svg.append("g")
        .attr("class", "y axis")
        .attr("transform", "translate("+width+",0)")
        .call(yAxis)
```

Step1：初始化

数据绑定

```
...
var data = [ 'one', 'two', 'three' ];

d3.select('body').selectAll('p')
    .data(data)
.enter().append('p')
    .text('Enter');
...
```

数据绑定

```
…
var data = [ 'one', 'two', 'three' ];

d3.select('body').selectAll('p')
    .data(data)
.enter().append('p')
    .text('Enter');
…
```

- **选择对象**
  - select
  - selectAll
- 绑定数据
- 数据与对象对应

数据绑定

```
...
var data = [ 'one', 'two', 'three' ];

d3.select('body').selectAll('p')
    .data(data)
.enter().append('p')
    .text('Enter');
...
```

- 选择对象
- **绑定数据**
  - datum

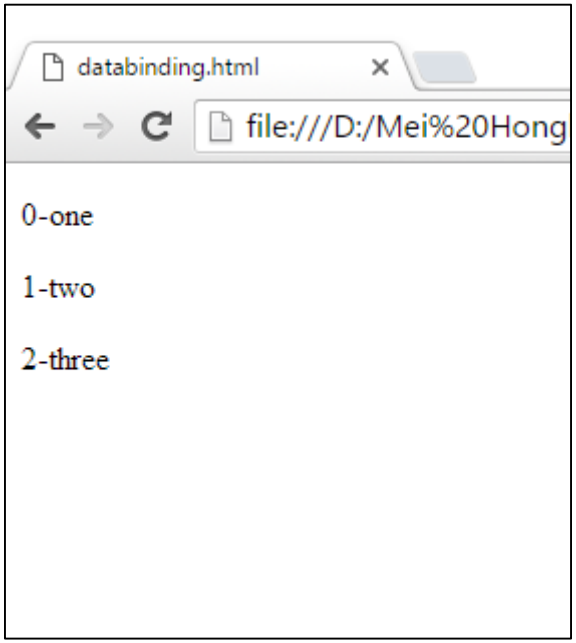
    绑定一个数据到选择集
  - data

    绑定一个数组到选择集，数组的各元素与选择集各元素分别绑定
- 数据与对象对应

数据绑定

```
…
var data = [ 'one', 'two', 'three' ];

d3.select('body').selectAll('p')
    .data(data)
.enter().append('p')
    .text(function(d,i){
        return i+'-'+d;
    });
…
```



数据绑定

```
…
.text(function(d,i){
        console.log(d,i);
        return i+'-'+d;
});
…
```

注：函数还有第三个参数，在嵌套选择时会被使用

数据绑定

```
…
var data = [ 'one', 'two', 'three' ];
d3.select('body').selectAll('p')
    .data(data)
.enter().append('p')
    .text('Enter');
…
```

- 选择对象
- 绑定数据
- **数据与对象对应**
  - Enter: 数据无对应对象
  - Update: 数据与对象对应
  - Exit: 对象无对应数据

数据绑定：三种子选择

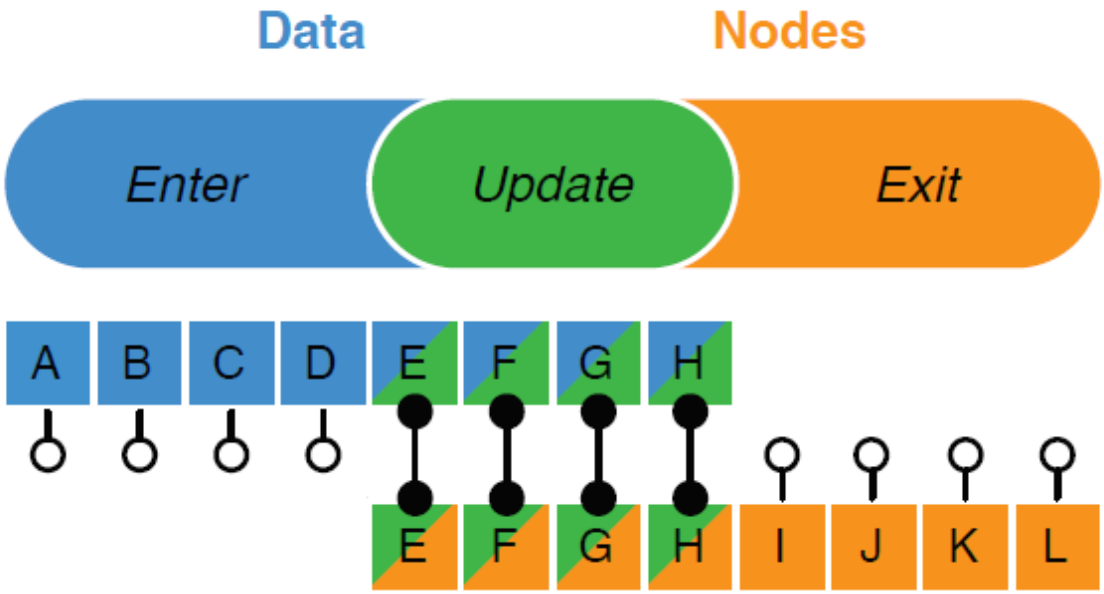


Fig. 6. When new data (blue) are joined with old nodes (orange), three subselections result: *enter*, *update* and *exit*.

数据绑定

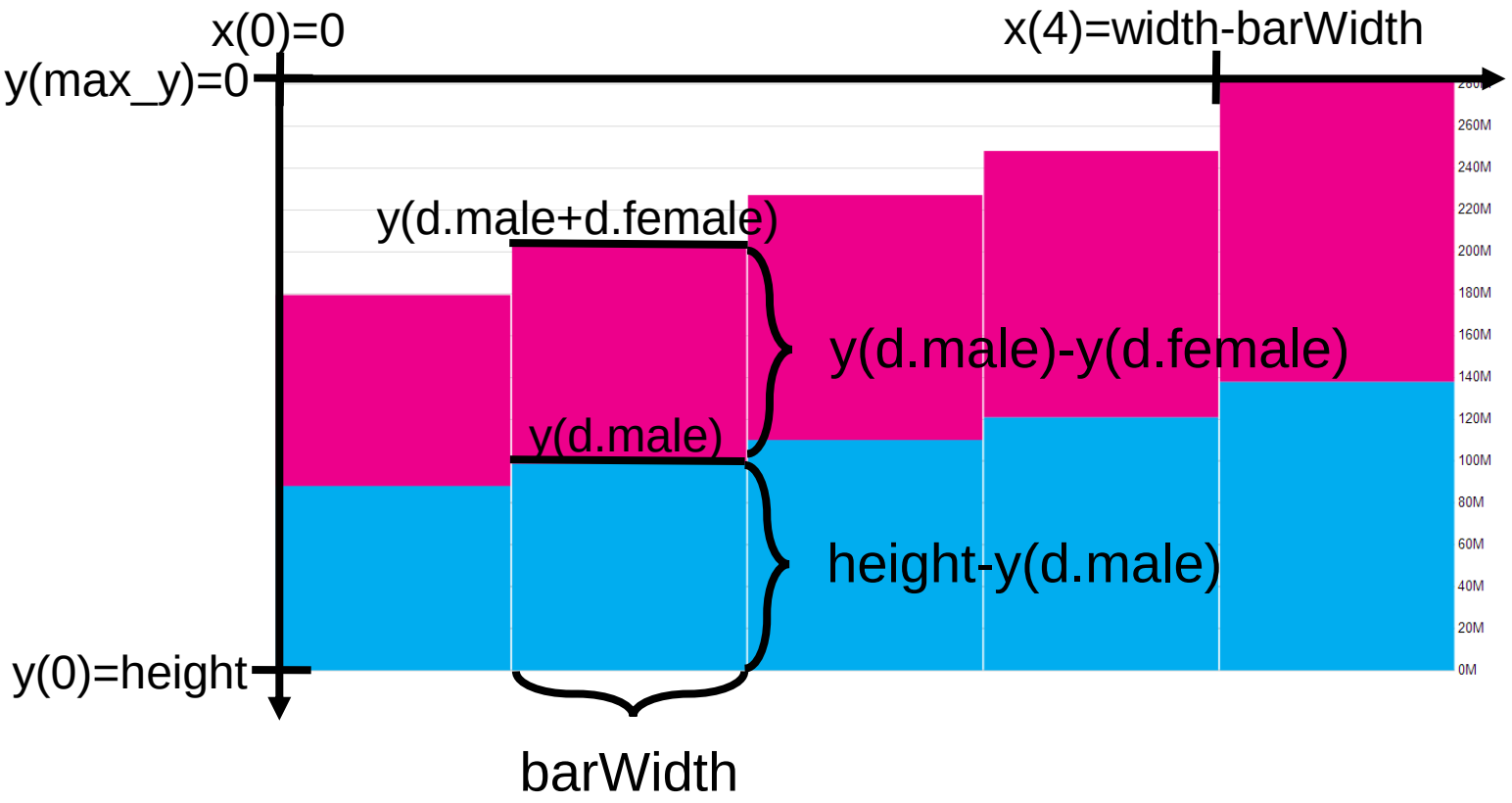
```
…
<p>Update</p>
…
var data = [ 'one', 'two', 'three' ];
var p = d3.select('body').selectAll('p')
    .data(data)
    .style('font-size','20px');
p.enter().append('p')
    .text('Enter');
p.style('color','red');
…
```

update

enter

update + enter

Step2：数据绑定


x(0)=0
x(4)=width-barWidth
y(max_y)=0
y(d.male+d.female)
y(d.male)-y(d.female)
y(d.male)
height-y(d.male)
y(0)=height
barWidth
260M
240M
220M
200M
180M
160M
140M
120M
100M
80M
60M
40M
20M
0M

Step2：数据绑定-Enter

```
/* male */
var bar_male=
        svg.selectAll('rect.male')
                .data(data);

bar_male.enter().append('rect')
        .attr('class','male')
        .attr('x',...
...
```

```
/* female */
var bar_female=
        svg.selectAll('rect.female')
                .data(data);

bar_female.enter().append('rect')
        .attr('class','female')
        .attr('x',...
...
```

Step2：数据绑定-Key

- Key函数
  - 与默认相同

```
.data(data, function(d, i){return i;})
```

  - 数据本身作为key

```
.data(data, function(d){return d;})
```

  - 数据的某一维度

```
.data(data, function(d){return d.id;})
```

```
var bar_male=svg.selectAll('rect.male')
                .data(data,function(d){return d.year;});
```

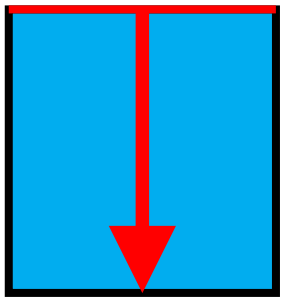
Step2：数据绑定-Update/Exit

```
//update
bar_male.attr('x',function(d,i){
        return x(i);
});
//exit
bar_male.exit().remove();
```

```
//update
bar_female.attr('x',function(d,i){
        return x(i);
});
//exit
bar_female.exit().remove();
```

Step3：动画

```
//exit
bar_male.exit().transition().duration(600)
        .attr('y',height)
        .attr('height',0)
.remove();
```



Step3：动画

```
//enter
bar_male.enter().append('rect')
        .attr('class','male')
        ...
//update
bar_male.transition().duration(600)
        .attr('x',function(d,i){return x(i);});
//exit
bar_male.exit()...
```

回顾：Update与Enter的顺序

```
…
<p>Update</p>
<script>
var data = [ 'one', 'two', 'three' ];
var p = d3.select('body').selectAll('p')
    .data(data)
    .style('font-size','20px');
p.enter().append('p')
    .text('Enter');
p.style('color','red');
</script>
…
```

update

enter

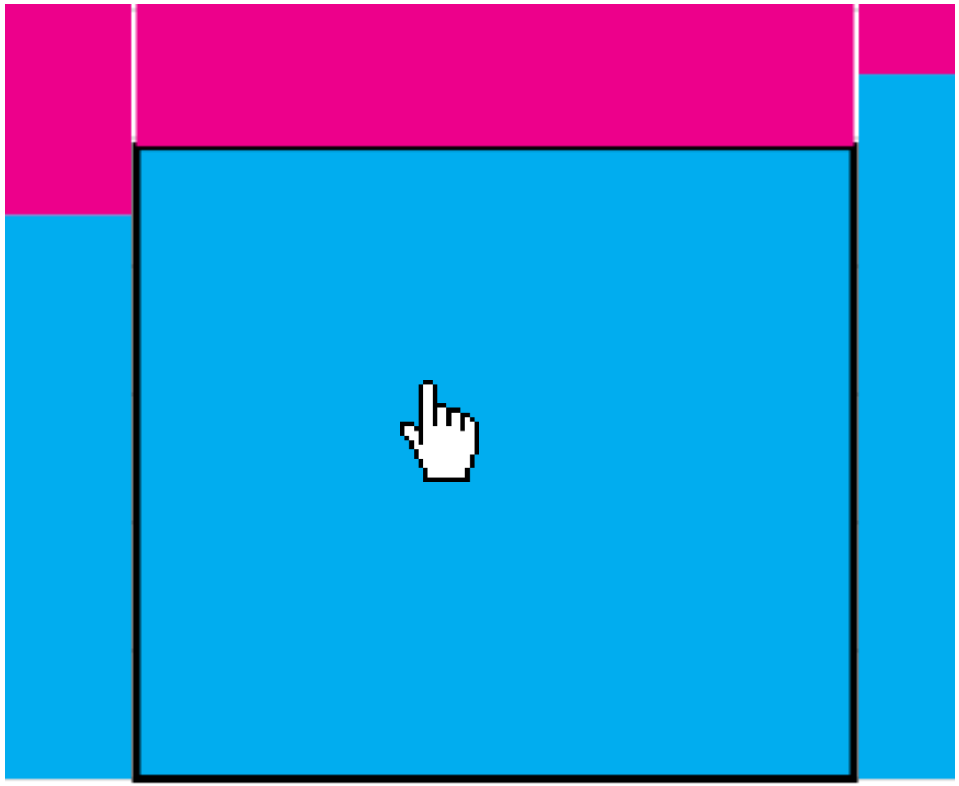
update + enter

- 注：v4中对Update不明确的问题做了改善，添加了Merge命令

Step4：交互-CSS

style.css:

```
rect:hover{
        cursor: pointer;
        stroke-width: 2px;
        stroke: black;
}
```



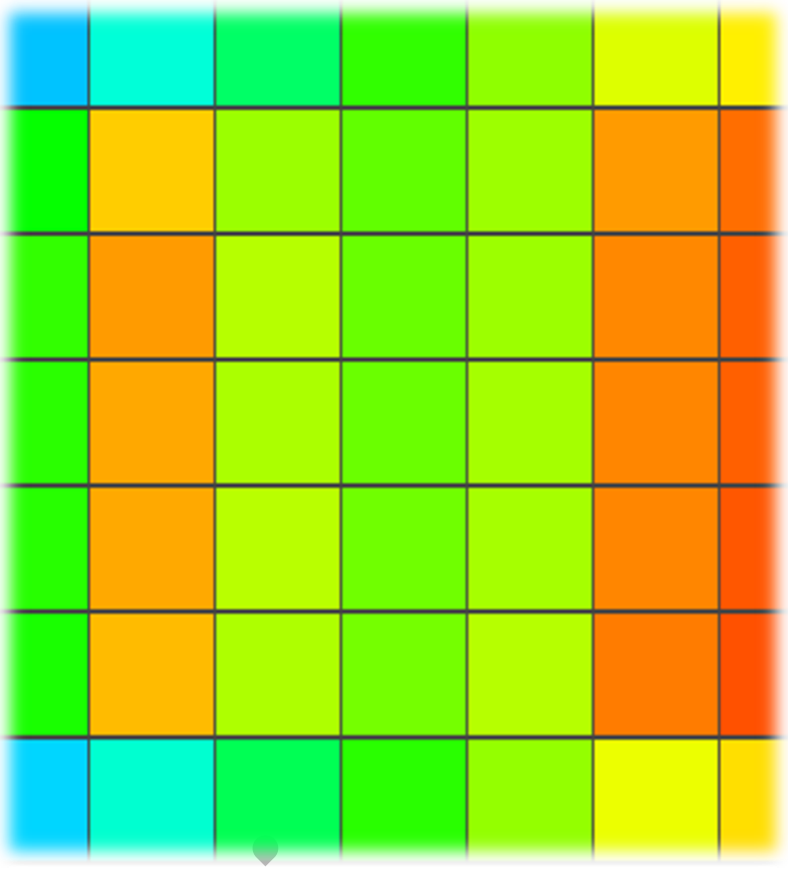
Step4：交互-鼠标响应

```
var onclick=function(d,i){
        console.log('clicked',i+'-'+d.year);
        …
}

bar_male.on('click',onclick);
bar_female.on('click',onclick);
```

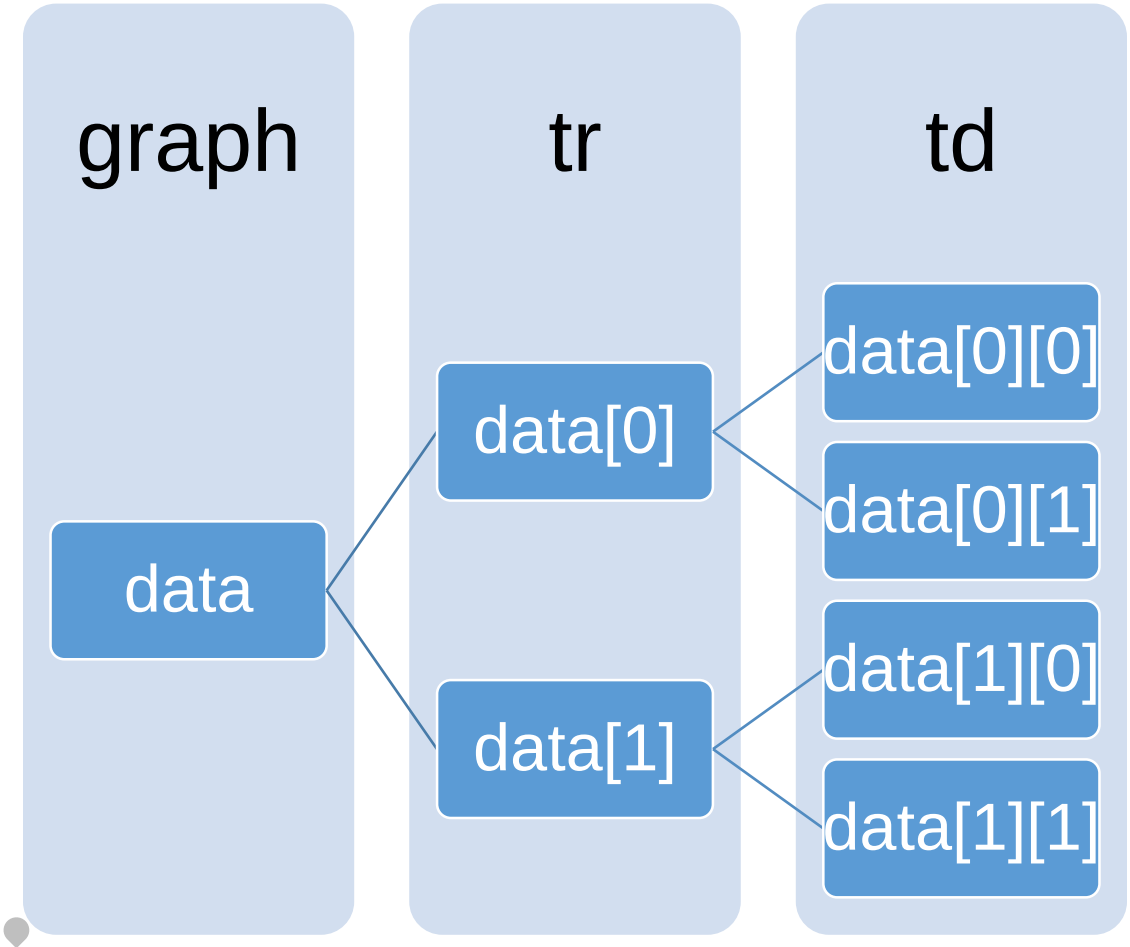
嵌套和布局 4

嵌套选择



```
▼<table id="graph">
  ▼<tr>
    ▶<td class="cell">…</td>
    ▶<td class="cell">…</td>
     ...
    </tr>
  ▶<tr>…</tr>
  ▶<tr>…</tr>
  ▶<tr>…</tr>
  ▶<tr>…</tr>
  ▶<tr>…</tr>
  ▶<tr>…</tr>
```

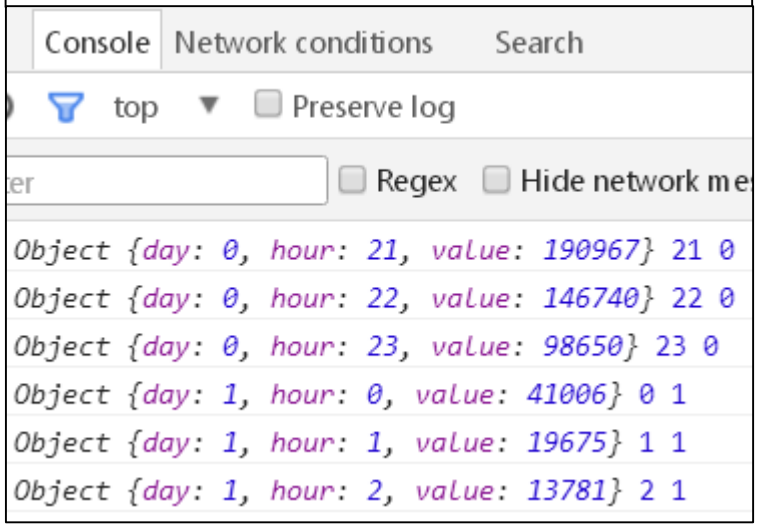
嵌套选择



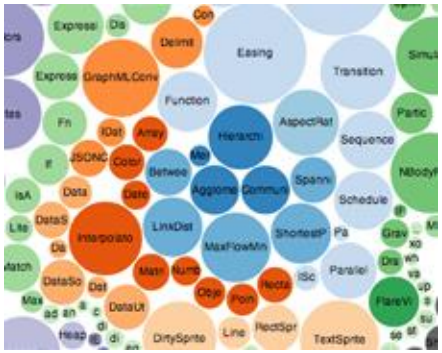
嵌套选择

```
var rows=d3.select("#graph")
    .selectAll("tr")
        .data(data)
    .enter().append("tr");
…
var cell=rows.selectAll("td")
        .data(function(d){return d;})
    .enter().append("td");
…
```

```
function(d,i,j){
        console.log(d,i,j);
}
```



d3.layout

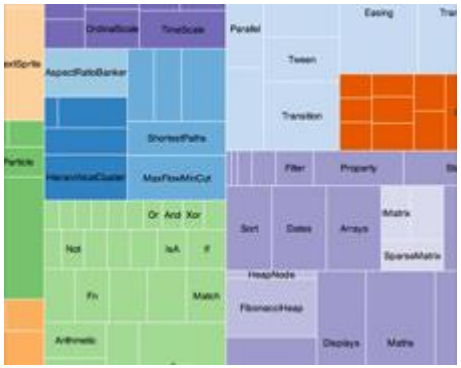


d3.layout.pack()

d3.pack()

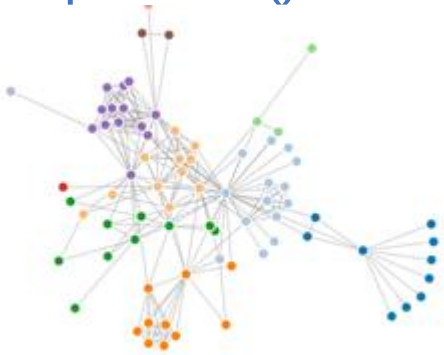
d3.layout.patition()

d3.partition()



d3.layout.treemap()

d3.treemap()



d3.layout.force()

d3.forceSimulation()

其他可视化工具 5

VEGA系列

```
{
  "name": "arc",
  "width": 400,
  "height": 400,
  "data": [
    {
      "name": "table",
      "values": [12, 23, 47, 6, 52, 19],
      "transform": [
        {"type": "pie", "value": "data"}
      ]
    }
  ],
  "scales": [
    {
      "name": "r",
      "type": "sqrt",
      "domain": {"data": "table", "field": "data"},
      "range": [20, 100]
    }
  ],
  "marks": [
    {
      "type": "arc",
      "from": {"data": "table"},
      "properties": {
        "enter": {
          "x": {"group": "width", "mult": 0.5},
          "y": {"group": "height", "mult": 0.5},
          "startAngle": {"field": "startAngle"},
          "endAngle": {"field": "endAngle"},
          "innerRadius": {"value": 20},
          "outerRadius": {"scale": "r"},
          "stroke": {"value": "#fff"}
        },
```

图的基本信息

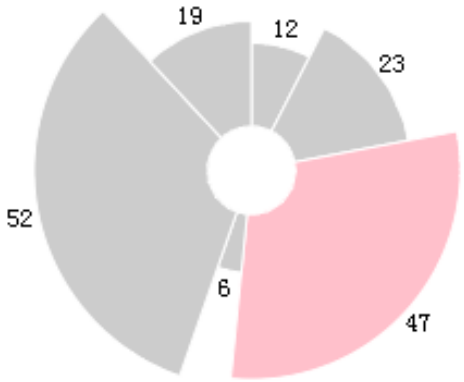
定义数据

数据变换

定义 Scale

定义图元

图元类型、数据来源、基本属性绑定

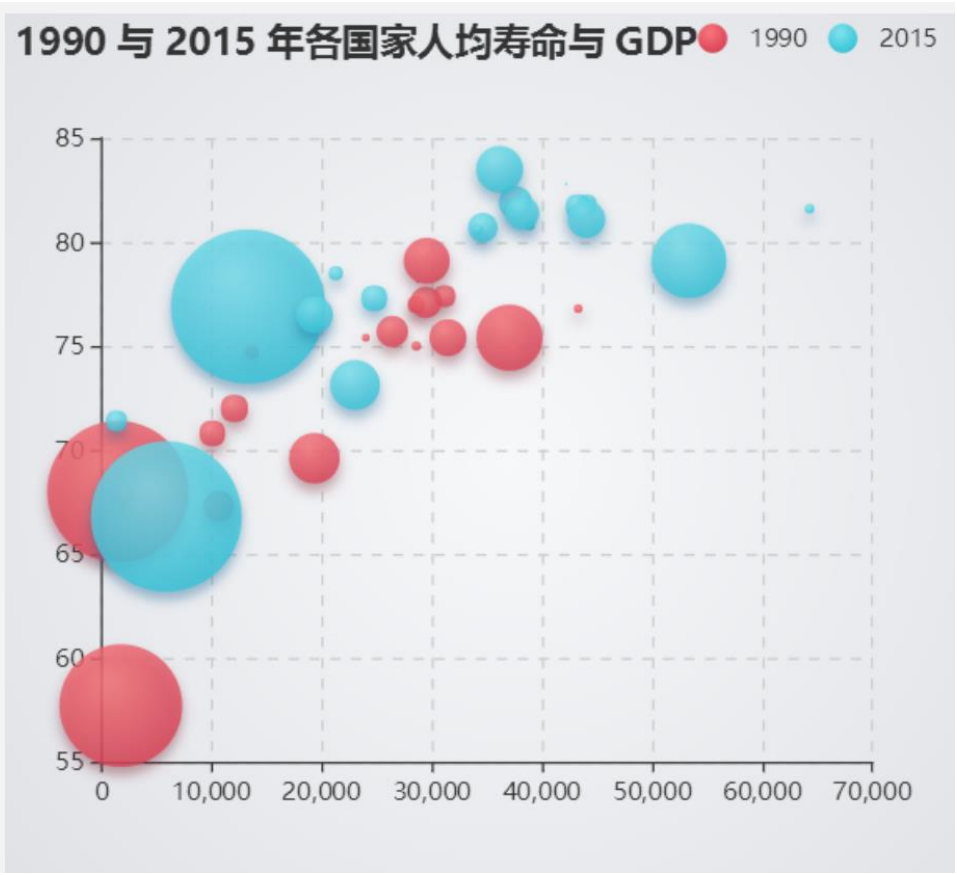


- VEGA
  - 基于D3之上的高层可视化规范语言(specification language)
- VEGA-Lite
  - 封装了VEGA图表作为模板
- Voyager
  - 基于VEGA-Lite，推荐图表

ECharts

16:34:22 图表已生成, 50ms 运行

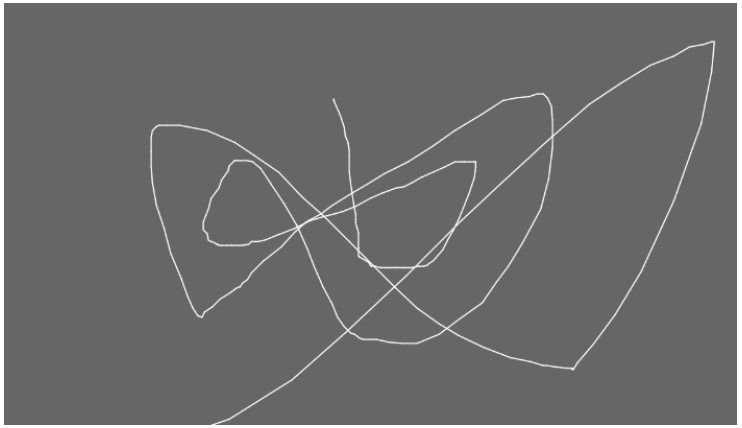
```
1  app.title = '气泡图';
2
3  var data = [
4      [[28604,77,17096869,'Australia',1990
5      [[44056,81.8,23968973,'Australia',20
6  ];
7
8  option = {
9      backgroundColor: new echarts.graphic
10         offset: 0,
11         color: '#f7f8fa'
12     }, {
13         offset: 1,
14         color: '#cdd0d5'
15     }]),
16     title: {
17         text: '1990 与 2015 年各国家人均
18     },
19     legend: {
20         right: 10,
21         data: ['1990', '2015']
22     },
23     xAxis: {
24         splitLine: {
25             lineStyle: {
26                 type: 'dashed'
27             }
28         }
```



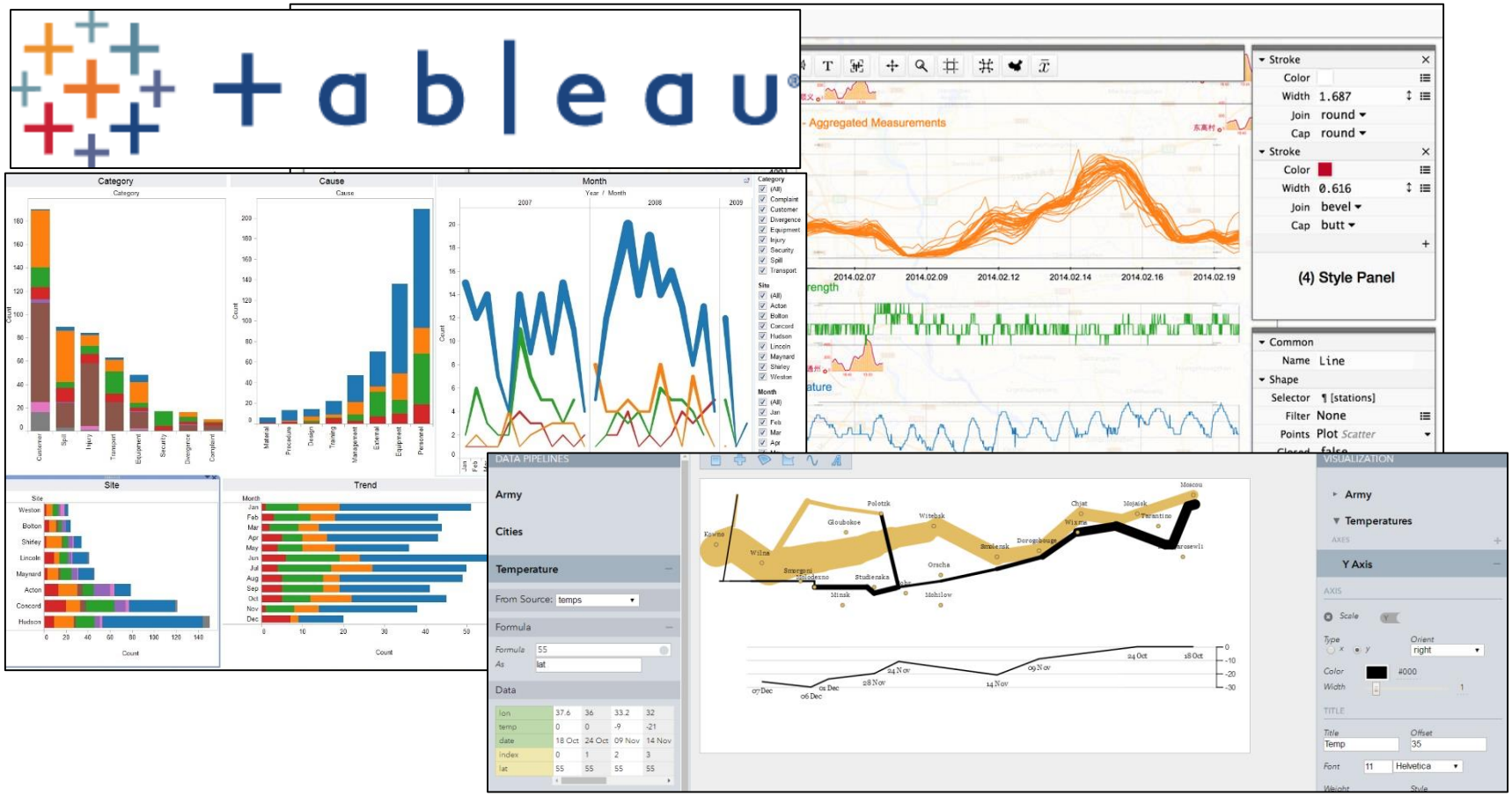
processing

```
void setup() {
  size(640, 360);
  background(102);
}

void draw() {
  stroke(255);
  if (mousePressed == true) {
    line(mouseX, mouseY, pmouseX, pmouseY);
  }
}
```



交互式工具


+ a b | e a u
Category
Cause
Month
Site
Trend
Aggregated Measurements
(4) Style Panel
DATA PIPELINES
Army
Cities
Temperature
From Source: temps
Formula
Data
VISUALIZATION
Army
Temperatures
Y Axis

可视化工具对比

- 学习成本
  - 交互式工具<ECharts<VEGA<D3<=processing
- 功能/灵活性
  - D3>=processing>VEGA>ECharts>交互式工具
- 性能效率
  - processing(桌面)>VEGA(canvas)=ECharts >D3=VEGA(SVG)>processing(web)

总结 6

总结

- D3是什么
- 为什么选择D3
- 如何构建一个D3程序
  - HTML+CSS+JavaScript+SVG
- 如何调试D3程序
  - 浏览器调试工具+命令行+console.log
- D3 API： https://github.com/d3/d3/blob/master/API.md
- D3 样例： https://github.com/d3/d3/wiki/Gallery

谢谢

联系方式

meihonghui.zju@gmail.com